MDM: Um Software Livre para Configuração de Ambientes Multiterminais

Aramis Stach Haiduski Fernandes, Francisco Panis Kaseker, Lucas Nascimento Ferreira, Paulo Ricardo Zanoni, Pedro Eugenio Rocha, Eduardo Todt*

*Departamento de Informática – Universidade Federal do Paraná

Caixa Postal 19.081 – CEP 81.531-980 – Curitiba – PR – Brasil

fashf03, fpk07, lnf07, paulo, pedro, todtg@c3sl.ufpr.br

Abstract. *One of the main problems in automatic hardware configuration for multiseat environments is the process of hardware detection and association. This process must be as automatic as possible, aiming to minimize the installation and maintenance costs. The devices present on a machine must be detected and associated to workstations, in a way that each one has at least a mouse, keyboard and monitor. This paper describes these problems and their main solutions, and also presents the Multiseat Display Manager (MDM), an open source tool that performs multiseat automatic configuration.*

Resumo. *Um dos principais problemas na configuração automática de ambientes multiterminais é o processo de detecção e associação de hardware. Este processo deve ser o mais automático possível, visando minimizar os custos de instalação e manutenção. Os dispositivos presentes em uma máquina devem ser detectados e associados a pontos de trabalho de forma que cada um possua ao menos um mouse, teclado e monitor. Este artigo descreve tais problemas e suas principais soluções, além de apresentar o Multiseat Display Manager (MDM), uma ferramenta de código aberto que realiza a configuração automática de multiterminais.*

1. INTRODUÇÃO

Um multiterminal ou *multiseat* é um computador que possui múltiplos dispositivos de entrada e saída, como monitores, mouses e teclados, agrupados em pontos de trabalho[1] independentes. Esse modelo possibilita o acesso ao computador para vários usuários simultaneamente [Oliveira *et al*. 2006]. Entre as vantagens no uso de multiterminais estão a maximização do uso de recursos computacionais, redução de custos e de consumo elétrico devido à economia de hardware, facilidade de manutenção e economia de pontos de rede. Essas vantagens são medidas em relação ao número de usuários por computador. Por exemplo, em um laboratório de informática que utiliza multiterminais, é possível atender a um mesmo número de usuários de um laboratório composto por monoterminais, porém utilizando menos computadores.

Atualmente existem diversas maneiras de criar ambientes multiterminais, todas elas devem levar em consideração que os computadores pessoais possuem uma estrutura desenhada para atender apenas um usuário. Como exemplo, em ambientes Linux, toda a entrada referente

[1] *Comumente chamado de estação de trabalho, cabeça, terminal, head ou seat.*

a mouses e teclados é diretamente repassada ao servidor de janelas *Xorg*, que as transmite às aplicações. Disso decorrem dois problemas fundamentais para uma solução multiterminal: detecção e associação dos dispositivos de entrada e saída. O problema de detecção consiste em identificar e ler cada dispositivo separadamente, para que suas entradas sejam repassadas somente às aplicações que devem recebê-las. A associação de dispositivos é o problema de agrupar os conjuntos de dispositivos em pontos de trabalho independentes.

Esse artigo apresenta o *Multiseat Display Manager* (MDM) [C3SL 2009], um software livre baseado em *Linux* e no *X Window System*. O MDM configura automaticamente um conjunto básico de dispositivos: mouses, teclados e placas de vídeo. Ele também dá suporte à implementação de diversas soluções multiterminais, como as baseadas em servidores de janelas aninhados (*Xnest, Xephyr*), múltiplas instâncias do *Xorg* dentre outras [Oliveira *et al*. 2006]. O MDM utiliza o conceito de módulos de execução para sua integração com as diversas soluções multiterminais. Os módulos definem funções de gerenciamento dessas soluções. Por exemplo, iniciar o servidor de janelas *Xorg* e as diversas instâncias de *Xephyr*.

O restante deste artigo está organizado da seguinte maneira: a seção 2 apresenta os problemas relacionados à detecção de dispositivos. A seção 3 apresenta os problemas relacionados com a associação de um conjunto de dispositivos em ambientes multiterminais. A seção 4 explica como o MDM soluciona os problemas apresentados e mostra como é a estrutura e a implementação de um módulo. Por fim, a seção 5 apresenta as considerações finais e os trabalhos futuros.

2. DETECÇÃO DE DISPOSITIVOS

Um dos desafios na configuração de ambientes multiterminais é detectar e separar corretamente os dispositivos de entrada e saída para que seja possível associá-los a estações de trabalho. Em ambientes monoterminais não há a necessidade de realizar uma detecção tão precisa, pois todos os dispositivos são diretamente atribuídos ao único usuário do sistema.

A detecção consiste em decidir quais dispositivos serão utilizados a partir de um conjunto de hardware já detectado pelo sistema operacional, obtendo as informações necessárias para, posteriormente, realizar a associação dos dispositivos.

Em ambientes Linux monoterminais, todas as entradas referentes aos mouses e teclados são diretamente repassados ao servidor de janelas *Xorg*, que as transmite às aplicações. Já em multiterminais cada dispositivo deve ser identificado e lido separadamente, pois é necessário descobrir se há um conjunto de dispositivos básicos para formar uma estação de trabalho.

Na configuração de multiterminais devem ser identificados ao menos os dispositivos básicos de uma estação de trabalho. O restante dessa seção explica como é feita a detecção em cada caso. O sistema também pode prover suporte a outros tipos de dispositivos, como *pendrives*, placas e caixas de som, dispositivos de armazenamento externos etc.

2.1. Dispositivos de vídeo:

Os dispositivos de vídeo são reconhecidos da mesma forma pelo Kernel Linux, independentemente do barramento utilizado (PCI, PCI *Express* ou AGP). Nestes ambientes, é possível obter a lista dos dispositivos de vídeo através do sysfs, que é um sistema de arquivos virtual que exporta informações sobre *drivers* e dispositivos de dentro do *kernel* para o espaço do usuário [Patrick 2005]. Como o *Kernel Linux* exporta informações sobre dispositivos da mesma forma, a detecção de dispositivos de vídeo pode abstrair o barramento utilizado pela placa.

Um problema exclusivo de dispositivos de vídeo é a associação entre hardware e *driver*. A maioria dos *drivers* de vídeo utilizados hoje em sistemas Linux faz parte do *Xorg*, e não do *kernel*. Por este motivo, o *Xorg* é responsável por fazer esta associação. Todo *driver* possui uma lista de dispositivos que suporta, utilizada pelo *Xorg* para descobrir a qual *driver* associar determinado dispositivo de vídeo. O *driver* fornece também informações sobre o número de saídas de vídeo que cada placa possui.

2.2. Dispositivos de Entrada:

Nem sempre é possível identificar com precisão a função de um dispositivo de entrada, pois as informações fornecidas pelo hardware podem não ser suficientes para identificá-la. Os dispositivos podem não reportar o que realmente são informando apenas como deve ser realizada a interação entre o hardware e o sistema [Zeuthen 2009]. Alguns teclados multimídia, por exemplo, possuem diversas funcionalidades e podem mostrar-se ao sistema operacional como sendo um conjunto de dispositivos de diversas funcionalidades. Em ambientes Linux, este problema é tratado associando as informações fornecidas pelo hardware a listas predefinidas de informações de dispositivos. Uma das principais ferramentas utilizadas para detecção é o *Hardware Abstraction Layer* (HAL) [Zeuthen 2009]. O HAL é um software que obtém informações do *kernel* e provê a descrição de todo o hardware presente no sistema a aplicações em modo usuário.

3. ASSOCIAÇÃO DE DISPOSITIVOS

Realizar a associação de hardware em ambientes multiterminal é agrupar os dispositivos existentes fazendo-os trabalhar em conjunto, formando pontos de trabalho. A partir do momento em que estão associados, esses dispositivos são destinados a atender um único usuário, independentemente dos outros dispositivos e pontos de trabalho presentes na máquina.

Um problema especial na associação de hardware são os dispositivos *hotplug*, que são os que podem ser conectados ao sistema sem necessitar reiniciá-lo, como *pendrives*, telefones celulares, MP3 *players*, mouses e teclados USB. Para reconhecer esses dispositivos são necessários mecanismos de associação que possibilitem que ela seja feita não somente de maneira estática, antes de configurar os pontos de trabalho, mas à medida que os dispositivos são inseridos e retirados da máquina.

Existem duas maneiras principais de realizar a associação de dispositivos: agrupar conjuntos de dispositivos em *hubs* USB, formando uma hierarquia USB e por *request/reply*.

3.1. Associação por Hierarquia USB

Uma solução adotada para realizar a associação de dispositivos é dividi-los por *hubs* USB. Cada *hub* USB é associado a uma placa de vídeo (monitor), formando um ponto de trabalho. Todos os dispositivos conectados em um determinado hub serão associados ao ponto de trabalho correspondente.

Esta abordagem simplifica a utilização de quaisquer dispositivos USB, como *pendrives*, placas de som USB e dispositivos *hotplug* em geral, permitindo que sejam associados diretamente ao ponto de trabalho que utiliza o hub USB ao qual o dispositivo foi ligado. Apesar disso, são necessários tantos *hubs* USB quantos pontos de trabalho, e a associação de outros tipos de dispositivos, como PS/2 ou PCI, não é tratada.

3.2. Associação por *request/reply*

Outra solução é a associação por *request/reply*. Neste tipo de associação, é exibido um *request* em cada monitor, por exemplo, o pedido de acionamento de determinada tecla ou botão do mouse. Cada monitor exibe um *request* diferente, de forma que ao receber o *reply*, enviado manualmente pelo usuário, o sistema possa reconhecer qual foi o monitor que o requisitou e associá-lo ao dispositivo.

Um dos problemas da associação por *request/reply* é que após configurado um ponto de trabalho, não é possível trocar, remover ou adicionar dispositivos já associados, a menos que o sistema forneça uma forma de retornar à interface de configuração. Esta abordagem não permite também a associação de determinados tipos de dispositivos, como placas de som, devido à dificuldade em implementar resposta do usuário para associação.

4. O MULTISEAT DISPLAY MANAGER

Multiseat Display Manager (MDM) [C3SL 2009] é um software que configura automaticamente computadores multiterminais. Ele foi elaborado para suportar diversos tipos de soluções multiterminais para Linux. Para tal suporte, oMDMutiliza módulos de execução que fazem a integração entre ele e o tipo de solução multiterminal utilizada.

A detecção de dispositivos de vídeo é feita através das informações *device id* e *vendor id* fornecidas pelos dispositivos. Esses identificadores são localizados em listas preexistentes, fornecidas pelo pacote de *software discover* [Stefan 2010], que informam quais *drivers* de vídeo suportam quais dispositivos. O problema desse método é que ele exige que as listas estejam corretas, e por isso será futuramente substituído por outro método que delegue toda a responsabilidade de detecção de *drivers* para o servidor de janelas *Xorg*.

O MDM detecta os dispositivos de entrada através do HAL; entretanto essa detecção não é completamente precisa. Um dispositivo físico com múltiplas funcionalidades, como por exemplo, um teclado multimídia, pode ser mostrado pelo HAL como sendo mais de um dispositivo de entrada. Esse tipo de dispositivo faz o MDM interpretar que há mais dispositivos que os realmente existentes no computador. Porém, detectar mais dispositivos de entrada não é um problema, pois o MDM considera que o número de pontos de trabalho é igual à quantidade de dispositivos de vídeo detectados, e não de mouses ou teclados.

A associação de dispositivos é feita pelo método *request/reply*. Esse método é vantajoso, pois ele independente de hardware adicional aos básicos de uma estação de trabalho. É importante também ressaltar que o MDM detecta e configura apenas mouses, teclados e placas de vídeo. Outros dispositivos como *pendrives*, placas de som e leitores de CD não são afetados pelo MDM.

Uma solução multiterminal deve, além de detectar e associar os dispositivos, prover uma forma de interação com os diversos usuários de maneira independente. O método utilizado consiste em abrir diversas telas de *login*, de um *Display Manager*, para cada usuário. Um *Display Manager* é um gerenciador gráfico de *login* e os mais populares são: *Gnome Display Manager* (GDM), *KDE Display Manager* (KDM) e o *X Display Manager* (XDM) [Jeff 2010]. O MDM não implementa um *Display Manager*, ele apenas utiliza os já existentes para abrir as diversas interfaces de *login* para os diferentes usuários. Devido à grande variedade de Display Managers, o MDM provê uma forma de integração (módulos) entre ele e os gerenciadores de *login*.

4.1. Módulos

As funções essenciais do MDM são detectar e associar os diversos dispositivos conec-

tados ao sistema para prover um ambiente multiterminal. O MDM não implementa um *Display Manager* para que os usuários façam *login* no sistema. Ele também não restringe de que forma será integrado o *Display Manager* ao ambiente gráfico do sistema. Por exemplo, o GDM pode ser executado em diversos *Xephyrs* sobre um mesmo servidor de janelas *Xorg*, ou sobre diversas instâncias do *Xorg* diretamente. Cada solução multiterminal utiliza métodos diferentes para prover uma interface de login no sistema. Para facilitar a integração com essas soluções, o MDM utiliza o conceito de módulos. Módulos são um conjunto de funções que disponibilizam ao usuário uma forma de fazer *login* através de um *Display Manager*. O módulo é quem define a forma como o *Display Manager* será integrado com o ambiente gráfico. Essa arquitetura do MDM facilita a inserção de novos módulos, pois basta escrever funções que implementam uma solução para o *Display Manager* desejado.

Para implementar um módulo é necessário criar no mínimo quatro funções:

1. **Display manager init:** essa função deve executar todas as tarefas anteriores ao início do *Display Manager*. Por exemplo, verificar se o *Display Manager* utilizado já está em execução.
2. **Display manager start monoseat:** inicia o *Display Manager* normalmente em caso do sistema possuir apenas uma placa de vídeo.
3. **Display manager start seat:** inicia os diversos servidores X com as telas de *login* do *Display Manager*. Caso o módulo utilize servidores X aninhados, essa função deve iniciá-los e iniciar as telas de *login* neles.
4. **Display manager stop:** termina a execução do *Display Manager* e, consequentemente, de todas as estações de trabalho.

Uma função adicional, chamada display manager start undeneath xserver, deve ser criada no caso do módulo utilizar servidores X aninhados. Essa função deve ser responsável por abrir o servidor X que os servidores aninhados vão utilizar como base para execução.

Atualmente, existem dois módulos implementados: xephyr-xdmcp e xephyr-gdm. O primeiro utiliza diversas instâncias do *Xephyr* para fazer conexões com um *Display Manager* remoto através do protocolo XDMCP. O XDMCP é um protocolo que provê um mecanismo uniforme para um display autônomo requisitar serviço de *login* a um hospedeiro remoto [Keith 2010a]. O módulo xehyr-gdm também utiliza instâncias do *Xephyr*, porém são abertas telas de *login* do GDM sobre cada *Xephyr*.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho analisa os problemas relacionados à detecção e associação automática de hardware em ambientes multiterminais, bem como descreve suas principais soluções. Além disso, foi apresentado o MDM, um software livre responsável por configurar automaticamente computadores multiterminais, implementando algumas das soluções apresentadas.

Devido a sua arquitetura modular, é possível desenvolver outros módulos para o MDM. Por exemplo, um módulo para uma solução multiterminal baseada em KDM e LTSP[2], para efetuar *login* em uma área de trabalho remota.

O servidor de janelas *Xorg*, em sua versão 1.8.0, passou a utilizar o *udev* como o mecanismo padrão para configuração automática de dispositivos de entrada [Keith 2010b]. O *udev*

2 *Linux Terminal Server Project*

é um software de detecção automática de dispositivos que provê um mecanismo dinâmico de nomeação dos mesmos. Dentre os trabalhos futuros está a substituição do HAL pelo *udev* como novo método de detecção de hardware. Essa alteração será feita para colocar o MDM nos novos padrões do *Xorg*.

O MDM utiliza soluções exclusivamente através de software para o processo de detecção e associação dos dispositivos. Esse tipo de solução multiterminal provê otimização de recursos de hardware e grande economia de energia. Portanto essa ferramenta pode ser muito interessante para o processo de inclusão digital.

REFERÊNCIAS

C3SL (2009). Multiseat Display Manager. http://wiki.c3sl.ufpr.br/multiseat/index.php/Mdm. Acessado em 14 de abril de 2009.

Jeff, C. (2010). GNOME Display Manager. http://live.gnome.org/GDM. Acessado em 03 de junho de 2010.

Keith, P. (2010a). X Display Manager Control Protocol. http://www.xfree86. org/current/xdmcp.pdf. Acessado em 03 de junho de 2010.

Keith, P. (2010b). Xorg 1.8.0 Release. http://lists.freedesktop.org/archives/xorg/2010-April/049784.html. Acessado em 03 de junho de 2010.

Oliveira, A. C., Vignatti, T., Weigaertner, D., Silva, F., Castilho, M., and Sunye, M. (2006). Um modelo de computação multiusuário baseado em computadores pessoais. *VII Workshop de Software Livre*, pages 135–140.

Patrick, M. (2005). The sysfs Filesystem. *Proceedings of the Linux Symposium*.

Stefan, K. (2010). Hardware Identification System. http://wiki.debian.org/discover. Acessado em 03 de junho de 2010.

Zeuthen, D. (2009). HAL 0.5.10 Specification. http://people.freedesktop. org/~david/hal-spec/hal-spec.html. Acessado em 14 de abril de 2009.